Structured Programming Language (CSE113)
Lecture #03
Loops in C programming

Loops are used for executing a block of statements repeatedly until a given condition returns false. There are three types of loop in C. They are:

1. ***for*** loop
2. ***while*** loop
3. ***do-while*** loop

1. *for* Loop

The ***for*** **loop** is commonly used when the number of **iterations** is known.
The syntax of for loop is:

```
for(initialization;  condition ;  increment/decrement)
{
  statement-block;
}
```

How *for* loop works?

- The **initialization** statement **is executed only once**.
- Then, the **condition (test expression)** is evaluated. If the **condition** is **false (0)**, *for* loop is terminated. But if the test expression is **true (nonzero)**, codes inside the body of ***for*** **loop** is executed and the **update expression (increment/decrement)** is updated.
- This process repeats until the **condition** is false.

<u>Flowchart of for loop</u>

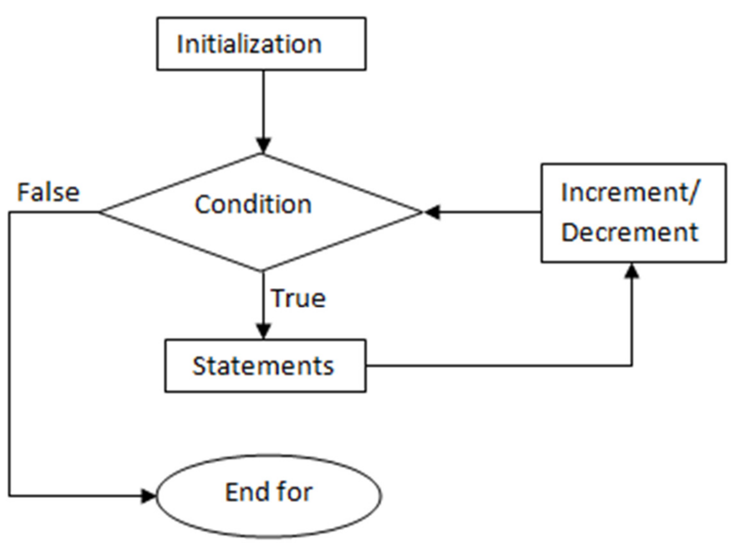


fig: Flowchart for for loop

Program Example 1: Display numbers from **1** to **10**

Source Code:

```
#include<stdio.h>
int main()
{
   int i;
   for(i = 1; i <= 10; i++)
      printf("%d\t",i);
   return 0;
}
```

Output:

```
"C:\Users\Md.Bulbul Ahamed\Desktop\C Programming\loop\loop.exe"
1       2       3       4       5       6       7       8       9       10

Process returned 0 (0x0)   execution time : 0.078 s
Press any key to continue.
```

Program Example 2: Display numbers from **1** to **n**

Source Code:

```
#include<stdio.h>
int main()
{
   int i,n;
   printf("Enter any numebr: ");
   scanf("%d",&n);  //Input upper limit from user
   for(i = 1; i <= n; i++)
      printf("%d\t",i);
   return 0;
}
```

Output:

```
"C:\Users\Md.Bulbul Ahamed\Desktop\C Programming\loop\loop.exe"
Enter any numebr: 20
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20

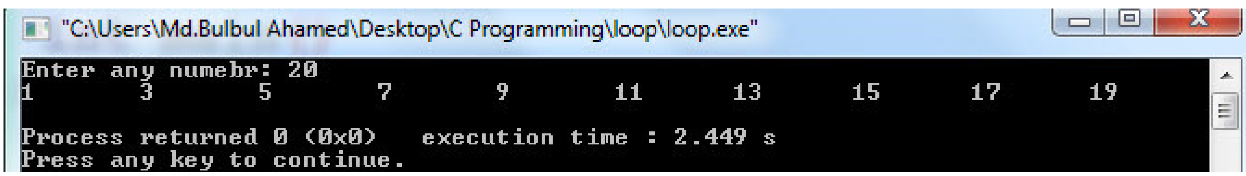
Process returned 0 (0x0)   execution time : 107.246 s
Press any key to continue.
```

Program Example 3: Display odd numbers from **1** to **n**

Source Code:

```
#include<stdio.h>
int main()
{
   int i,n;
   printf("Enter any numebr: ");
   scanf("%d",&n);  //Input upper limit from user
   for(i = 1; i <= n; i = i + 2)
      printf("%d\t",i);
   return 0;
}
```

Output:

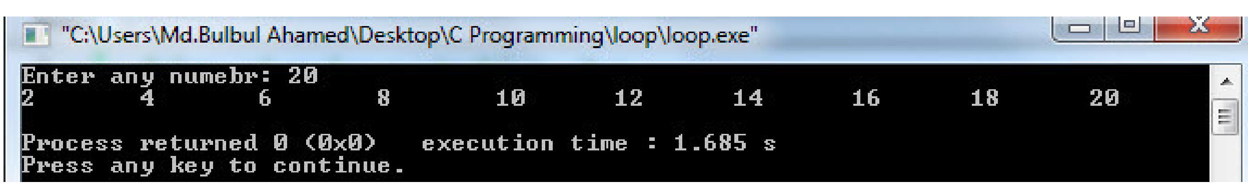


Program Example 4: Display even numbers from **1** to **n**

Source Code:

```
#include<stdio.h>
int main()
{
   int i,n;
   printf("Enter any numebr: ");
   scanf("%d",&n);  //Input upper limit from user
   for(i = 2; i <= n; i = i + 2)
      printf("%d\t",i);
   return 0;
}
```

Output:

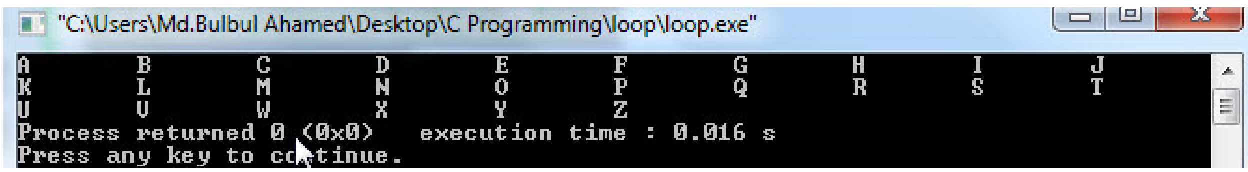


Program Example 5: Display alphabets from **A** to **Z**

Source Code:

```
#include<stdio.h>
int main()
{
   char i;
   for(i = 'A'; i <= 'Z'; i = i + 1)
      printf("%c\t",i);
   return 0;
}
```

Output:

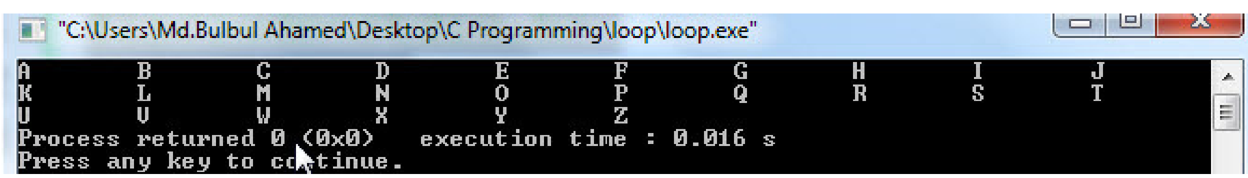


Program Example 6: Display alphabets from **A** to **Z** using ASCII values

Source Code:

```
#include<stdio.h>
int main()
{
   int i;
   for(i = 65; i <= 90; i = i + 1)
      printf("%c\t",i);
   return 0;
}
```

Output:

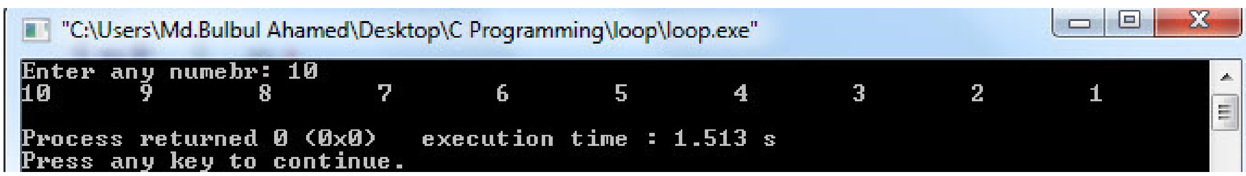


Program Example 7: Display number from **n** to **1**

Source Code:

```
#include<stdio.h>
int main()
{
   int i,n;
   printf("Enter any numebr: ");
   scanf("%d",&n); //Input upper limit from user
   for(i = n; i >= 1; i = i - 1)
      printf("%d\t",i);
   return 0;
}
```

Output:

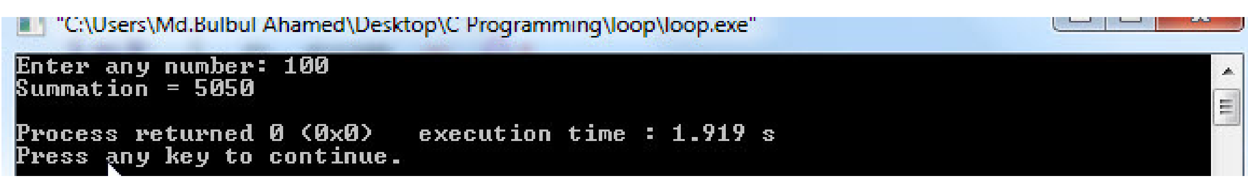


Program Example 8: Sum of number from **1** to **n**

Source Code:

```
#include<stdio.h>
int main()
{
   int i,n,sum = 0;
   printf("Enter any number: ");
   scanf("%d",&n); //Input upper limit from user
   for(i = 1; i <= n; i = i + 1){
      sum = sum + i;
   }
      printf("Summation = %d\n",sum);
   return 0;
}
```

Output:

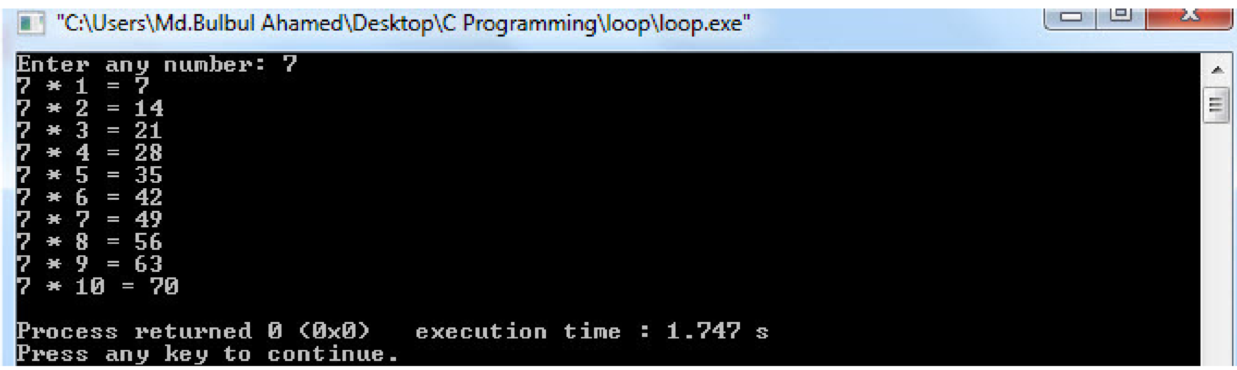


Program Example 9: Display multiplication table

Source Code:

```
#include<stdio.h>
int main()
{
   int i,num,p;
   printf("Enter any number: ");
   scanf("%d",&num); //Input upper limit from user
   for(i = 1; i <= 10; i = i + 1){
      p = num * i;
    printf("%d * %d = %d\n",num,i,p);
   }
   return 0;
}
```

Output:



2. *While* loop

- **while** loop iterates the code until condition is false. Here, condition is given before the code. So code may be executed 0 or more times.
- It is better if number of iteration is not known by the user.

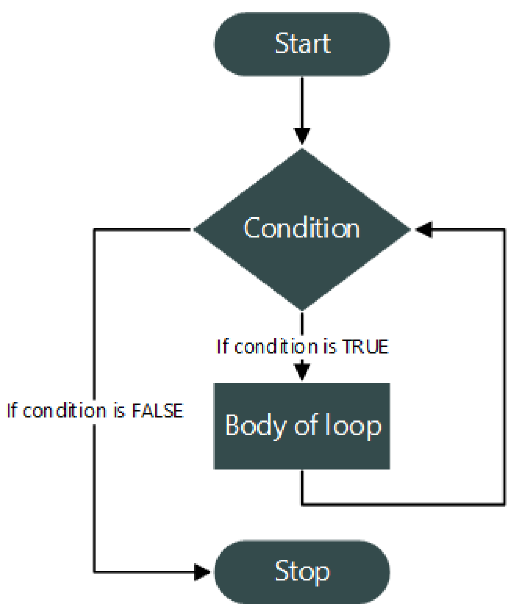
The syntax of a while loop is:

```
while (condition)
{
    //Statements to be executed repeatedly
}
```

How while loop works?

- The while loop evaluates the **condition**.
- If the **condition** is true (nonzero), codes inside the body of while loop are executed. The **condition** is evaluated again. The process goes on until the **condition** is false.
- When the **condition** is false, the while loop is terminated.

Flowchart of *while* loop



Infinite while loop

If the condition to be tested is always true, the loop will run forever i.e. infinite times.

For example,

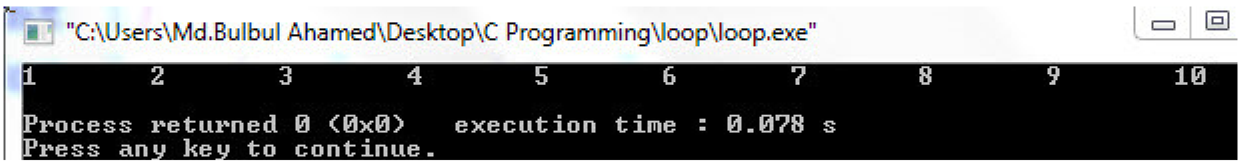
```
while (1)
{
   printf("This is infinite loop");
}
```

Program Example 1: Display numbers from **1** to **10**

Source Code:

```
#include<stdio.h>
int main()
{
   int i = 1;
   while(i <= 10){
      printf("%d\t",i);
      i++;
   }
   return 0;
}
```

Output:



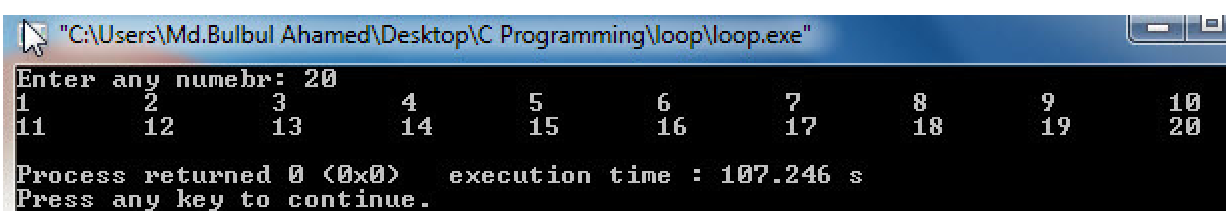
```
"C:\Users\Md.Bulbul Ahamed\Desktop\C Programming\loop\loop.exe"
1       2       3       4       5       6       7       8       9       10
Process returned 0 (0x0)   execution time : 0.078 s
Press any key to continue.
```

Program Example 2: Display numbers from **1** to **n**

Source Code:

```
#include<stdio.h>
int main()
{
   int i = 1,n;
   printf("Enter any number: ");
   scanf("%d",&n);
   while(i <= n){
      printf("%d\t",i);
      i++;
   }
   return 0;
}
```

Output:



```
"C:\Users\Md.Bulbul Ahamed\Desktop\C Programming\loop\loop.exe"
Enter any numebr: 20
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20
Process returned 0 (0x0)   execution time : 107.246 s
Press any key to continue.
```

3. *do…while* loop

- In **do while** loop, statement is given before the **condition**, so statement or code will be executed at **least one time**. In other words, we can say it is executed 1 or more times.
- It is better if you have to execute the code at least once.

The syntax of C language do-while loop is given below:

```
do{

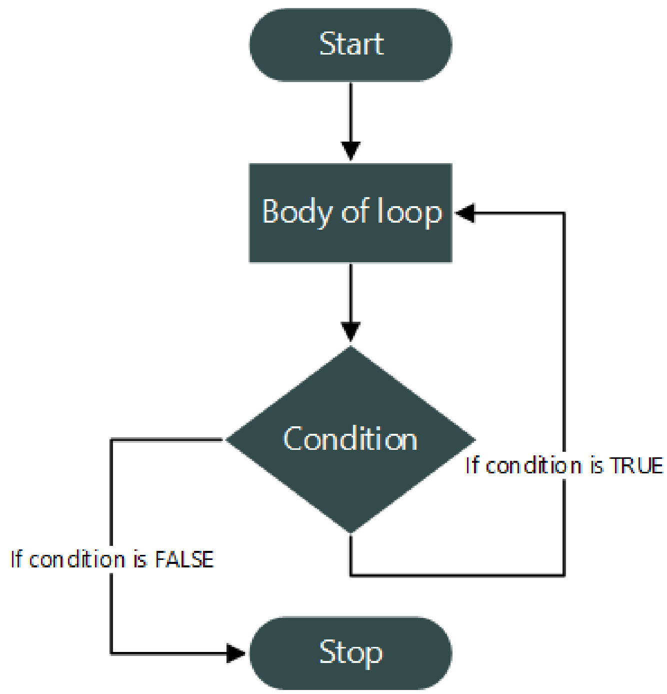
    //code to be executed

} while(condition);
```

How do…while loop works?

- The code block (loop body) inside the braces is executed once.
- Then, the **condition** is evaluated. If the **condition** is true, the loop body is executed again. This process goes on until the **condition** is evaluated to 0 (false).
- When the **condition** is false (nonzero), the do...while loop is terminated.

<u>Flowchart of *do...while* loop</u>



Infinite do-while loop

There may be a condition in a do-while loop which is always true. In such case, the loop will run infinite times. **For example**,

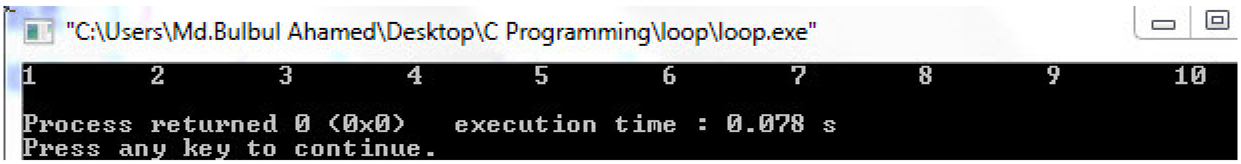
```
do
{
   printf("This is infinite loop");
}while(1);
```

Program Example 1: Display numbers from **1** to **10**

Source Code:

```
#include<stdio.h>
int main()
{
   int i = 1;
   do{
      printf("%d\t",i);
      i++;
   }while(i <= 10);
}
```

Output:

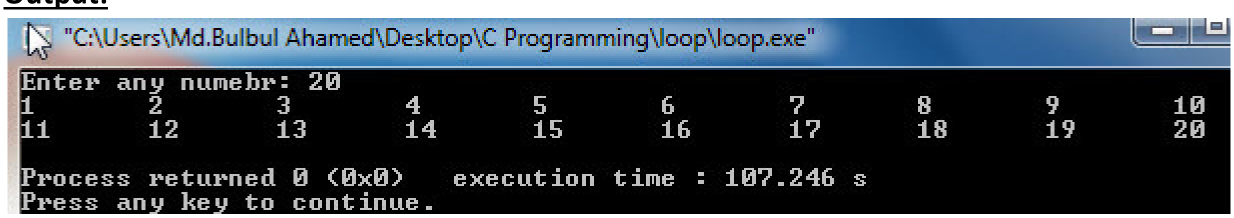


Program Example 2: Display numbers from **1** to **n**

Source Code:

```
#include<stdio.h>
int main()
{
   int i = 1,n;
   printf("Enter any number: ");
   scanf("%d",&n);
   do{
      printf("%d\t",i);
      i++;
   }while(i <= n);
}
```

Output:



Difference between while & do while loops in C language:

while	do while
Loop is executed only when condition is true.	Loop is executed for first time irrespective of the condition. After executing while loop for first time, then condition is checked.